Received 31 January 2024; revised 16 May 2024; accepted 20 May 2024. Date of publication 27 May 2024; date of current version 1 July 2024. The review of this article was arranged by Associate Editor Lisandro Lovisolo.

Digital Object Identifier 10.1109/OJSP.2024.3405719

Adaptable L4S Congestion Control for Cloud-Based Real-Time Streaming Over 5G

JANGWOO SON (Associate Member, IEEE), YAGO SANCHEZ, CORNELIUS HELLGE, AND THOMAS SCHIERL (Senior Member, IEEE)

Fraunhofer Heinrich Hertz Institute, 10587 Berlin, Germany

CORRESPONDING AUTHOR: JANGWOO SON (email: jangwoo.son@hhi.fraunhofer.de).

This work was supported by the German Federal Ministry of Education and Research as part of the VoluProf Project under Grant 16SV8705.

ABSTRACT Achieving reliable low-latency streaming on real-time immersive services that require seamless interaction has been of increasing importance recently. To cope with such an immersive service requirement, IETF and 3GPP defined Low Latency, Low Loss, and Scalable Throughput (L4S) architecture and terminologies to enable delay-critical applications to achieve low congestion and scalable bitrate control over 5G. With low-latency applications in mind, this paper presents a cloud-based streaming system using WebRTC for real-time communication with an adaptable L4S congestion control (aL4S-CC). aL4S-CC is designed to prevent the target service from surpassing a required end-to-end latency. It is evaluated against existing congestion controls GCC and ScreamV2 across two configurations: 1) standard L4S (sL4S) which has no knowledge of Explicit Congestion Notification (ECN) marking scheme information; 2) conscious L4S (cL4S) which recognizes the ECN marking scheme information. The results show that aL4S-CC achieves high link utilization with low latency while maintaining good performance in terms of fairness, and cL4S improves sL4S's performance by having an efficient trade-off between link utilization and latency. In the entire simulation, the gain of link utilization on cL4S is 1.4%, 4%, and 17.9% on average compared to sL4S, GCC, and ScreamV2, respectively, and the ratio of duration exceeding the target queuing delay achieves the lowest values of 1% and 0.9% for cL4S and sL4S, respectively.

INDEX TERMS Cloud computing, congestion control, L4S, mixed reality, real-time communication.

I. INTRODUCTION

With the emerging interest in interactive multimedia communication in immersive services, low-latency streaming for seamless interaction has been gaining importance. Examples thereof are the increasing deployment of delay-sensitive immersive services such as Microsoft xCloud [1], Meta Horizon Workrooms [2], and NVIDIA CloudXR [3]. These deployed services rely on cloud-based immersive systems that offload complex graphics processing like volumetric video and 3D gaming rendering to an edge server, which provides interactive video streaming with low latency. Thereby, such solutions aim at solving the issue that currently deployed client devices still have limitations in computing and rendering complex realistic representations [4].

In line with the emergence of the delay-sensitive services, 3rd Generation Partnership Project (3GPP), a united organization of standard organizational partners to develop protocols for mobile telecommunications, has defined terminologies in the 5G system architecture [5] to support the new congestion control (CC) service called by Low Latency, Low Loss, and Scalable Throughput (L4S) [6]. L4S is designed to achieve low queue delay, loss, and jitter with a scalable rate adaptation by utilizing Explicit Congestion Notification (ECN) [7] bits on the IP header. The general mechanism of L4S components is 1) the network scheduler marks Congestion Experienced (CE) code point on the ECN field to signal that the network is suffering from congestion, 2) the client aggregates the number of CE-marked packets and received packets, and sends the

© 2024 The Authors. This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

congestion information to the server, and 3) the server adjusts the rate by applying the received congestion information to the CC algorithm.

The experiments within this paper focus on a cloud-based interactive streaming service [8] that utilizes L4S and meets the 5G criteria. L4S has been integrated into our system using Web Real-Time Communication (WebRTC) due to the following reasons: 1) the WebRTC open source is already widely used for real-time communication and is compatible with various devices (e.g. browsers, android, IOS, and others), and 2) the high-level architecture and syntaxes of Real-time Transport Protocol (RTP) on the 5G network for XR services are being discussed based on WebRTC through 3GPP Technical Specification (TS) [9]. In addition, we focus mainly on streaming using 5G networks, in which the bottleneck typically occurs between the base station and the User Equipment (UE). In this paper, L4S components on the cloud-based streaming system are described, and an adaptable L4S-CC (aL4S-CC) is presented, as an extension of [10], that achieves a high link utilization with low latency while being tolerant against uncertain network conditions (e.g. variability of link capacity). In order to evaluate its performance, the proposed aL4S-CC has been compared with the Google Congestion Control (GCC) [11] and Self-Clocked Rate Adaptation for Multimedia (ScreamV2) [12].

The remainder of the paper is structured as follows. Section II discusses the status of CC on Real-time Communication (RTC) with a description of GCC and ScreamV2 and explains improvements over our previous work [10]. Subsequently, Section III presents L4S features integrated into the cloud-based streaming system, and Section IV describes the proposed aL4S-CC. Then, the performance of aL4S-CC is evaluated in comparison to GCC and ScreamV2 using metrics mandated by immersive services in Section V. Finally, in Section VI, we highlight the conclusion of this paper.

II. RELATED WORK

CC has been a crucial research topic since 1988 [13], and numerous CCs in Transmission Control Protocol (TCP) have been proposed: the window-based CC like TCP-Vega [14], TCP-Cubic [15], and Copa [16] adapt the window size derived from the congestion metrics; the rate-based CC like PCC [17], TCP-BBR [18] adjust the transmission rate for fast re-synchronization to reduce frame delay. Recently, CC in TCP specialized for cloud services has been proposed [19], [20].

However, CC has not only been investigated for TCP. Due to the increasing traffic for delay-sensitive RTC, several studies have investigated CC for real-time streaming.

GCC [11], existing CC in WebRTC, adjusts the encoding bitrate to the minimum value that is derived by two controllers as explained in the following: 1) a loss-based controller that modifies the encoding bitrate using three different intervals, namely a 0–2% packet loss interval, in which the bitrate is increased, a 2–10% packet loss interval during which the encoded bitrate is kept as is and the case with a packet loss higher than 10% for which the encoded bitrate is decreased, with the specific bitrate being computed based on the actual packet loss ratio received from a feedback message via Real-Time Transport Control Protocol (RTCP); 2) a delay-based controller that estimates the queuing delay through a Kalman filter using an inter-group delay variation derived from RTCP and determines the rate according to whether over or under-utilization of the link is detected by comparing the queuing delay with a delay threshold.

Next, ScreamV2 [12] supports L4S in RTP and adjusts the rate based on the ECN marking and the congestion window. More concretely, the congestion window prevents excessive amounts of data from being injected into the network when the link drops rapidly, and the algorithm mainly considers three parameters (i.e. CE-marking ratio, Round Trip Time (RTT), and Maximum Segment Size (MSS)) through the two CE marks per RTT rule as given by

$$\text{target rate} = (2.0/pMark) * MSS * 8/RTT\ [bps] \tag{1}$$

where $pMark$ is a packet marking probability.

In addition, the learning-based CC in RTP has emerged recently. Loki [21] presents a hybrid solution that integrates rule-based (i.e. GCC) and learning-based CCs through feature-level fusion, and LCC [22] employs the learning-based CC into WebRTC by training a probability density model through a one-way delay and sending rate. They improve reliable transmission with low latency but still involve unavoidable overhead for supplementary implementation and training, along with limitations for universal deployment. In this regard, CC based on L4S is oriented toward the criterion in the 5G standard and takes advantage of explicit signaling for queuing delay.

Another CC in RTP, our prior work [10], has been developed for low-latency applications leveraging L4S and is herein referred to as L4S-CC. It utilizes ECN marking as the main indicator of congestion. More concretely, when packets are not CE- marked, three phases are defined based on the duration being CE-marking ratio as 0%; 1) slight decrease until 300 ms; 2) gradual increase from 300 ms to 1 s; 3) aggressive increase over 1000 ms. Otherwise, when CE-marked packets are encountered, the target rate is decreased in proportion to the CE-marking ratio from 13% to 64%/sec pace through constant parameters, and in the case of the CE-marking ratio being 100%, the rate is sharply dropped in order to prevent the target End-to-end (E2E) delay from being surpassed. Thus, the results demonstrate the achievement of stable streaming with low latency by inferring queuing delay from ECN marking while upholding link utilization and preventing latency spikes. However, as depicted in Fig. 1, which illustrates Jain's fairness [23] of L4S-CC and the proposed standard L4S (sL4S) simulated on the IETF pattern (defined in Section V) and a constant link capacity of 10 Mbps, L4S-CC suffers from fairness when two flows with the same CC share the link, which should be improved. Also, the constant parameters used by the algorithm are not appropriate for the ability to cope with variations of link capacity and indicate that the

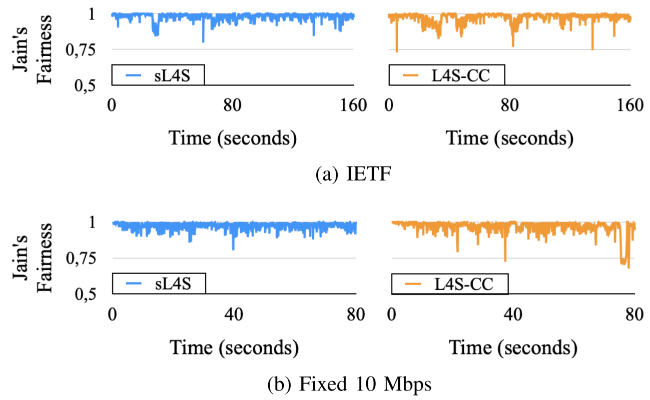


FIGURE 1. Jain's fairness of sL4S and L4S-CC (pattern: *IETF* and fixed 10 Mbps, target E2E delay: 60 ms, low threshold: 5 ms, high threshold: 20 ms).

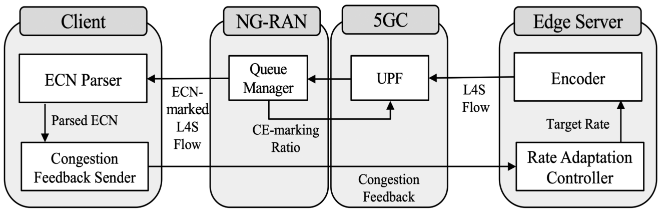


FIGURE 2. L4S flow on cloud-based streaming over 5G.

algorithm has been optimized for a particular implementation of the ECN marking scheme as discussed in [10], namely no CE-marked packets with a queuing delay below 4 ms (low threshold) and all packets being CE-marked for a queuing delay larger than 14 ms (high threshold). Accordingly, the proposed aL4S-CC has been developed to cope with different ECN marking schemes, enhancing compatibility with L4S for XR services by quickly reacting against the throughput variations and distributing the link fairly while keeping the good performance of L4S-CC as described in Section IV.

III. L4S ON CLOUD-BASED STREAMING

In this section, we describe the L4S mechanism in cloud streaming system that adheres to the 5G standard as illustrated in Fig. 2.

A. L4S-CAPABLE 5G NETWORK

The ECN marking for L4S is supported in two alternative ways: 1) the scheduler (i.e. Fair Queuing Controlled Delay (FQ-Codel) [24] and Dual-Queue Coupled Active Queue Management (AQM) [25]) on Next Generation Radio Access Network (NG-RAN) directly marks the CE code point on the IP header for each packet depending on the queue congestion level, 2) NG-RAN sends the ratio of CE-marked packets to PDU Session Anchor User Plane Function (PSA UPF) to perform ECN marking instead. The 5G system architecture on release 18 version [5] states that the criteria processing ECN marking is based on NG-RAN or UPF implementation specific, which means the ECN marking scheme (e.g. low and high thresholds of CE-marking) is not pre-defined but can be chosen arbitrarily by each implementation, e.g. determined by each operator. Accordingly, to evaluate coping ability with different Packet Delay Budget (PDB) [5] over 5G, targeting low latency for cloud-based real-time service, three cases of two thresholds (low and high) have been chosen for the experiments carried out in this paper, namely case 1 (2 ms and 10 ms), case 2 (5 ms and 20 ms), and case 3 (20 ms and 50 ms). The ECN marking is performed as follows by the network: if the queuing delay is the low threshold or less, no packets are CE-marked; if the queuing delay is the high threshold or more, all packets are CE-marked; if the queuing delay lies between the two thresholds, a percentage of the packets are CE-marked, with this percentage being computed linearly proportional to the queuing delay relative to the threshold interval (e.g. when the queuing delay is 4 ms and 8 ms in case 1, the probability of CE marking is 25% and 75%, respectively).

B. 5G-XR CLIENT

It receives the media stream for which the ECN marking was performed via *L4S-capable 5G Network*. The ECN marking information for each packet is interpreted in WebRTC low socket API which is modified to extract ECN two bits on the IP header (i.e. ECN field access through *setsockopt* function). In case the packet is CE-marked, the client accumulates the number of CE-marked packets. Then, the client specifies the accumulated number of CE-marked packets and the number of all received packets in a feedback message and sends the feedback to the server. The counting of the CE-marked and all received packets is initialized after sending the feedback message, and the frequency of sending the feedback message depends on the video frame interval according to the criterion defined in [26].

C. 5G-XR EDGE SERVER

The proposed aL4S-CC belongs to the server and derives the scalable rate to minimize latency and increase link efficiency. Thus, the encoder on the server adjusts the encoding bitrate to the target rate which is estimated from aL4S-CC using the received feedback message, and paces the packets over time to prevent congestion caused by an overflowed burst. The details of aL4S-CC are described in the following section.

IV. ADAPTABLE L4S CONGESTION CONTROL

In this section, we present the proposed aL4S-CC, which aims to achieve high link utilization with low latency while maintaining good performance of fairness. Compared to L4S-CC, we improved adaptability against volatile networks by estimating a delay gradient and utilizing the ECN marking scheme. In the following, the mechanism for each component of aL4S-CC, as shown in Fig. 3, and their contributions to achieving the mentioned goals are described.

A. CONGESTION TRACKER

As discussed in Section II, the parameters of L4S-CC are biased towards specific network environments. Accordingly,

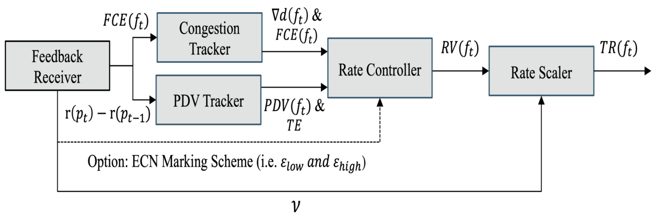


FIGURE 3. **aL4S-CC architecture: main elements and parameters.**

this component aims to facilitate efficient link utilization with low latency by adapting to unpredictable network conditions (e.g. wireless signal strength, link distribution, and scheduling). In such circumstances, the rate should be decreased aggressively in the case of extreme congestion to prevent delays or increased in the case of sufficient link capacity to improve link utilization. On the other hand, the rate needs to be maintained when approaching the link capacity boundary while keeping a stable latency. Accordingly, the algorithm adjusts a delay gradient $\nabla d(f_t)$ by inferring network fluctuations through congestion trend σ and duration τ for which no CE-marked packets have been received as shown in the following.

$$\begin{cases} \nabla d(f_t) = \nabla d(f_{t-1}) + \sigma & \text{if } 0 < \mathrm{FCE(f_t)} \text{ and } 0 < \sigma \\ \nabla d(f_t) = \nabla d(f_{t-1}) - \tau & \text{if } 0 = \mathrm{FCE(f_t)} \end{cases} \tag{2}$$

where f_t is a t-th feedback message, FCE is a fraction of CE-marked packets divided by received packets, and σ is given by

$$\sigma = \sum_{f_k \in \mathbb{Z}} FCE(f_k) - FCE(f_{k-1}) \tag{3}$$

where $\mathbb{Z}$ is a set of feedback messages belonging to the history window. Thus, the algorithm increases $\nabla d(f_t)$ depending on how quickly the queue becomes congested ($\propto \sigma$) by probing the congestion history and, on the other side, decreases $\nabla d(f_t)$ in proportion to the estimated queue availability ($\propto 1/\tau$). We set $\nabla d(f_t)$ to a range from 0.5 to 4.5 to prevent excessive in/decrease.

B. PDV TRACKER

It measures Packet Delay Variation (PDV), which is the difference between the arrival time intervals of two packet groups ($r(p_t)$ and $r(p_{t-1})$) compared to the sending time intervals of the same two groups ($s(p_t)$ and $s(p_{t-1})$) as

$$PDV(f_t) = (r(p_t) - r(p_{t-1})) - (s(p_t) - s(p_{t-1})) \tag{4}$$

PDV is used to seize the increment of E2E delay (i.e. encoding delay, queuing delay, processing delay, and transmission delay between end devices) severity in order to avoid suffering over the target E2E delay required by services.

C. RATE CONTROLLER

It infers the encoding rate through the congestion information received from *Congestion Tracker*, *PDV Tracker*, and *Feedback Receiver*. Depending on CE-marking intensity, the rate is adjusted as the following parts.

1) CE-MARKING RATIO = 0%

In this part, the algorithm measures the duration τ for which no CE-marked packets have been received. Here, we modified two aspects of L4S-CC described in Section II. On the one side, the gradual increase phase from 300 to 1000 ms was identified as being too aggressive and affecting fairness as shown in Fig. 1. On the other side, the slight decrease right after congestion when the ECN marked ratio is equal to 0 was identified to be unnecessary and detrimental. Therefore, the three phases, in which the non-CE marked case was subdivided, are substituted by the following equation

$$RV(f_t) = (\alpha(f_t))^{\tau} \tag{5}$$

where RV is a rate variation ratio which is a variable to adjust a target rate TR in (14), and $\alpha(f_t)$ is given by

$$\alpha(f_t) = \delta + \nabla d(f_t) \cdot k \tag{6}$$

where

$$\nabla d(f_t) \cdot k \in [-0.02 \text{ to } 0.02] \tag{7}$$

and δ is the initial value for a base $\alpha(f_t)$ and k is a variable for scaling $\nabla d(f_t)$, and $\nabla d(f_t) \cdot k$ ranges from -0.02 to 0.02 in proportion to $\nabla d(f_t)$. We empirically set δ to 1.05, $\alpha(f_t)$ is a range from 1.03 to 1.07 depending on $\nabla d(f_t)$. When τ is greater than 0 (i.e. right after a congestion phase finishes), RV is increased exponentially depending on the base $\alpha(f_t)$ and exponent τ. This ensures that for small values of τ, the increase of RV is minimal, preventing the congestion caused by aggressive rate increases and maintaining fairness among other streams, while still efficiently probing for available rate increases.

2) 0% < CE-MARKING RATIO < 100%

In case the CE-marking ratio is greater than 0 (i.e. congestion is detected), the algorithm decreases the rate to react against the congestion. To avoid unnecessary reductions in the early stages of congestion, the rate decrease is proportional to the CE-marking ratio. Another aspect of this proportionality is to suppress growing congestion. For this reason, L4S-CC has proven to provide stable streaming while refraining from latency spikes, but the decrease rate is only dependent on the ECN ratio, which leads to difficulty in coping with situations that require adaptation due to fluctuating network environments (e.g. changes in queue buffer length). Therefore, we integrate $\nabla d(f_t)$ into this part as

$$RV(f_t) = -\nabla d(f_t) \cdot FCE(f_t) \cdot \gamma \tag{8}$$

where $\nabla d(f_t)$ copes with the fluctuation as described in Sections IV-A and γ is a weight which is adjusted depending on the target E2E delay and the suffered delay (e.g. queueing delay). Furthermore, this equation enhances fairness by applying a differential $d(f_t)$ for each flow (e.g. flow experiencing congestion with large $d(f_t)$ undergoes a substantial bitrate reduction, narrowing the bitrate gap with other flow). γ is set to 1 as default when the algorithm doesn't know the ECN marking scheme (i.e. marking mechanism, low and high thresholds), and it is different from 1 when recognizing the ECN marking scheme (e.g. the scheme is transmitted from NG-RAN to end device via Radio Resource Control (RRC) and Non-access Stratum (NAS) protocol). In the latter provision, the algorithm is able to estimate a numeric queuing delay from FCE and two thresholds, and RV is revised depending on γ derived from the following equation.

$$\begin{cases} \begin{array}{c} \gamma = (\epsilon_{high} - \epsilon_{low})/(TQ - \epsilon_{low}) \\ \text{and } \max(\gamma) = 10 \end{array} & \text{if } \epsilon_{\text{low}} < \text{TQ} < \epsilon_{\text{high}} \\ \begin{array}{c} \gamma = \epsilon_{high}/TQ \\ \text{and } \min(\gamma) = 0.1 \end{array} & \text{if } \epsilon_{\text{high}} < \text{TQ} \\ \begin{array}{c} \gamma = 10 + (\epsilon_{low}/TQ) \\ \text{and } \max(\gamma) = 20 \end{array} & \text{otherwise} \end{cases} \tag{9}$$

where ϵ_{low} and ϵ_{high} are the low and high threshold for the ECN marking scheme, and γ is a range from 0.1 to 20 depending on the maximum target queuing delay TQ and its relation to the two thresholds. Note that TQ is the highest delay permissible during packet transmission while still meeting a specific target E2E delay for a particular service. TQ considers the queuing delay among the delays occurring in E2E, as the description to define TQ in the experiment in Section V.

3) CE-MARKING RATIO = 100%

When the available throughput suddenly drops unexpectedly, for instance in cases such as signal interference or sudden increase of users, the algorithm sharply decreases the rate to avoid large unacceptable latency spikes. Besides, such intense congestion has a significant impact on increasing E2E delay. Therefore, the algorithm focuses on preventing the defined target E2E delay that is acceptable on the specific service from being surpassed, and the higher PDV close to the target E2E delay TE leads to a sharper rate decrease. Hence, we define n to drive how fast the desired TR drop according to

$$\begin{cases} n = 3 \text{ and } PDV(f_t) = \eta & \text{if PDV}(\text{f}_\text{t}) <= \eta \\ n = \left\lceil \frac{TE - PDV(f_t)}{TE} \cdot 3 \right\rceil & \text{if } 0 < \text{PDV}(\text{f}_\text{t}) < \text{TE} \\ n = 1 & \text{otherwise} \end{cases} \tag{10}$$

where n is needed to be determined from categorized values, so low threshold η of TE is defined by

$$\eta = \frac{10}{TE} \tag{11}$$

and finally RV is determined by

$$RV(f_t) = \left(\left(\frac{\eta}{PDV(f_t)} \right)^{1/n} - 1 \right) \cdot \nu \tag{12}$$

where

$$\min\left(\frac{RV(f_t)}{\nu} \right) = 0.95 \tag{13}$$

and ν is a feedback interval used in (14) to scale TR through the estimated RV from each feedback message. Equation (12) multiplies ν, so at least 95% of TR is dropped in one feedback message, and the higher PDV is, the smaller n is, leading to a sharper bitrate decrease.

D. RATE SCALER

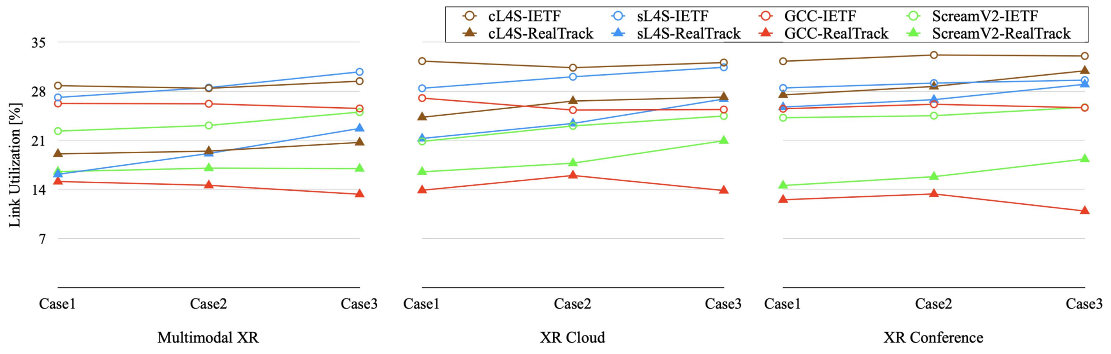
From a general aspect, the CC works better on the more frequent feedback message, but the interval needs to be limited to prevent excessive overhead. Accordingly, the feedback message interval is set to once per frame [26] and has a deviation within tens of milliseconds. To achieve a smooth rate change, *Rate Controller* scales the estimated rate to make up for the feedback interval deviation and derives the target rate TR as

$$TR(f_t) = TR(f_{t-1}) + \left(\frac{TR(f_{t-1}) \cdot RV(f_t)}{\nu} \right) \tag{14}$$

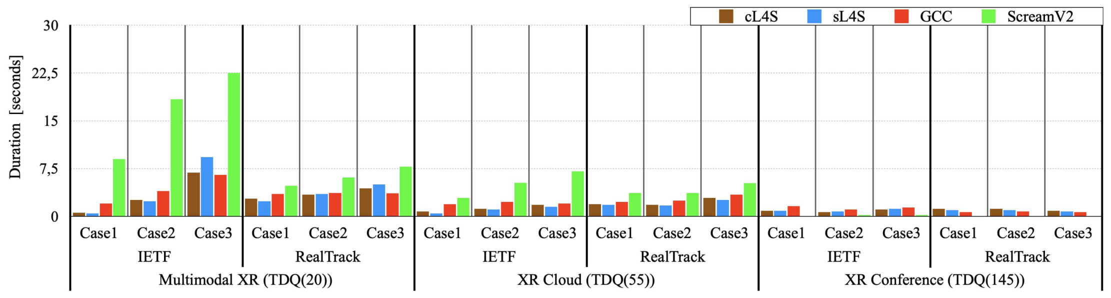
where $RV(f_t)$ determines how much $TR(f_t)$ is changed, and ν is the feedback interval time between f_t and f_{t-1} to scale $TR(f_{t-1}) \cdot RV(f_t)$ for each the size of ν.

V. EVALUATION

As the given knowledge in Section IV, we differentiate two cases for the proposed CC algorithm (i.e. one knowing and the other one not knowing the ECN marking scheme) and evaluate aL4S-CC comparing it to L4S-CC, GCC (tag version: 2.4.0 on Unity WebRTC github [27]) and ScreamV2 (commit version: 56c7abe on ScreamV2 github [28]). Therefore, aL4S-CC has been classified into two types; 1) standard L4S (sL4S) which doesn't have the ECN marking scheme information, and 2) conscious L4S (cL4S) which recognizes the ECN marking scheme information. Also, in order to analyze the suitability of the proposed algorithm with respect to meeting the Quality of Service (QoS) on different immersive use cases, the experiment targets to evaluate its performance based on three XR services having different required E2E delays according to standardized guidelines as follows; 1) service type: multimodal XR, required E2E delay: 25 ms [29], 2) XR cloud gaming (XR cloud), 60 ms [30], and 3) XR video conference (XR conference), 150 ms [5]. Accordingly, the target E2E delay (TE in (10)) on cL4S and sL4S has been set to as above, along with the maximum target queuing delay (TQ in (9)). Note that since the processing and propagation delay are considered to be very low in the considered scenario, with measurements in our network simulation and encoding delay of around five milliseconds, the value of TQ used in the algorithm is computed by subtracting 5 ms from the target



(a) Link utilization



(b) TDQ

FIGURE 4. **Link utilization and TDQ measured under sharing the link with heterogeneous traffic.**

E2E delay (i.e. 20 ms, 55 ms, and 145 ms for each service respectively).

Next, the emulation setup represents real-world experiments. The testbed is based on the cloud-based streaming system [8] into which aL4S-CC, L4S-CC, GCC, and ScreamV2 have been integrated. The L4S-capable network simulator [31] imitates a base station with the ECN marking capability and has been run based on three cases of two thresholds (i.e case 1 (2 ms and 10 ms), case 2 (5 ms and 20 ms), and case 3 (20 ms and 50 ms)) as described in Section III. Here, we assume case 3 on multimodal XR is out of target in terms of L4S implementation on the service provider because the target E2E delay (i.e. 25 ms) is too low compared to the two thresholds to react against congestion based on ECN marking. Accordingly, the result values on average presented in this paper exclude the above case. Although this is not the scope of the paper, it is pointed out that a form of communication is potentially established to appropriately configure the thresholds, ensuring the requirements of a specific service are met, thereby resolving the described issue.

In order to evaluate whether aL4S-CC meets the QoS required for XR services, the experiment has been conducted in three scenarios: 1) inter-protocol on multi-flows; 2) intra-protocol on multi-flows; 3) single-flow. Each scenario is simulated on *IETF* pattern which is based on evaluation criteria [32] for interactive real-time media discussed in IETF with the addition of linear in/decrease patterns and an expanded link capacity more suitable for the experiments as shown in Fig. 5. Furthermore, to evaluate the ability to cope with fluctuating networks in the presence of different congestion control algorithms, *RealTrack* pattern, measured from real-world traces using a mobile device over a 5G network, is employed additionally in scenario 1).

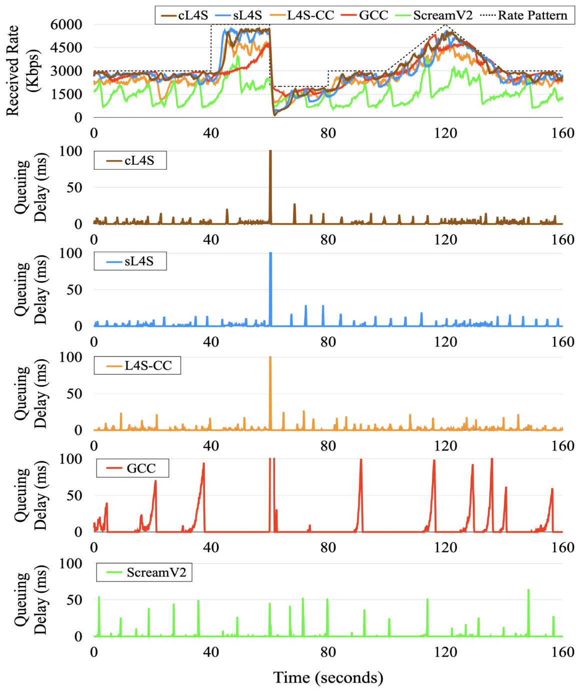


FIGURE 5. **Received rate and queuing delay measured under single flow (pattern: *IETF*, marking scheme: case 2, service type: XR cloud gaming).**

TABLE 1. Results on cL4S, sL4S, L4S-CC, GCC, and ScreamV2 Under Sharing the Link With Two Traffic Having Same CC

		Multimodal XR			XR Cloud			XR Conference				
	Metrics	cL4S	sL4S	L4S-CC	cL4S	sL4S	L4S-CC	cL4S	sL4S	L4S-CC	GCC	ScreamV2
	Link Utilization (%)	39.7	39.3	38	40.4	40	38.6	40.3	39.6	38.9	39.2	29.8
	QD Percentile 95% (ms)	2	2	2	2	2	2	2	2	2	7	1
	QD Percentile 99% (ms)	9	5	4	6	4	5	11	9	7	37	21
Case 1	TDQ(20) (seconds)	0.9	0.6	0.5	-	-	-	-	-	-	2.6	1.9
	TDQ(55) (seconds)	-	-	-	0.4	0.4	0.4	-	-	-	1.4	0.3
	TDQ(145) (seconds)	-	-	-	-	-	-	0.2	0.5	0.2	1.3	0
	Jain's Fairness	0.97	0.97	0.97	0.98	0.98	0.97	0.98	0.98	0.96	0.97	0.96
	Link Utilization (%)	40.2	40.6	38.4	41	40.7	39.4	41.3	40.4	40.6	38.7	31.9
	QD Percentile 95% (ms)	5	6	4	6	6	5	8	6	6	5	4
	QD Percentile 99% (ms)	15	16	11	15	15	14	35	33	32	49	33
Case 2	TDQ(20) (seconds)	0.9	1.2	1.1	-	-	-	-	-	-	3	2.9
	TDQ(55) (seconds)	-	-	-	0.5	0.4	0.5	-	-	-	1.4	0.4
	TDQ(145) (seconds)	-	-	-	-	-	-	0.4	1.3	0.5	1.2	0
	Jain's Fairness	0.97	0.97	0.97	0.98	0.98	0.97	0.98	0.98	0.96	0.98	0.95
	Link Utilization (%)	35.2	38	35.9	40.3	39.3	39	41.1	39.5	39.7	38.6	34.9
	QD Percentile 95% (ms)	18	20	21	23	22	31	34	24	30	8	22
	QD Percentile 99% (ms)	41	47	49	50	48	63	66	54	59	56	55
Case 3	TDQ(20) (seconds)	7.6	8.1	9.3	-	-	-	-	-	-	2.6	8.2
	TDQ(55) (seconds)	-	-	-	1.2	1.1	2.5	-	-	-	1.5	0.9
	TDQ(145) (seconds)	-	-	-	-	-	-	0.4	0.9	0.6	1.4	0.1
	Jain's Fairness	0.89	0.93	0.92	0.97	0.97	0.96	0.97	0.97	0.96	0.97	0.92

A. INTER-PROTOCOL COMPETITION

In this scenario, we evaluate the performance to provide stable media streaming in the presence of traffic transmitted with the other CCs. The results on Fig. 4(a) and (b) are measured for the case that three flows with aL4S-CC (cL4S or sL4S), GCC, and ScreamV2 are simultaneously being run over the same link, and (a) depicts link utilization which is a ratio of the received rate in the link capacity on each *IETF* (circle) and *RealTrack* (triangle), and (b) represents Total Duration over Queuing Delay (TDQ(λ)) which is an accumulated time whenever the measured Queuing Delay (QD) (i.e. time that a packet waits in the queue) is larger than λ on the two patterns. Thus, in order to compare the performance of cL4S and sL4S as well as comparing aL4S-CC itself with GCC and ScreamV2, each cL4S (brown) and sL4S (blue) have been run alternately together with GCC (red) and ScreamV2 (green) (i.e. first group: cL4S, GCC, and ScreamV2, second group: sL4S, GCC, and ScreamV2). The results of GCC and ScreamV2 on the two alternative simulations had the same behavior with subtle differences, so the average values of the two simulations are shown in Fig. 4, respectively. As can be seen in the results, GCC demonstrates relatively better results in a stable network pattern (i.e. *IETF* pattern). ScreamV2 shows a low TDQ range of 0 to 200 ms in the XR video conference, which assumes that it is optimized for a specific service by employing a relatively constant rate of change in (1). In contrast, aL4S-CC utilizing target E2E in (10) generally achieves the best trade-off between having a high link utilization and low latency for all cases. As intended in (9), cL4S has a similar performance to sL4S on case 2 of multimodal XR due to high threshold (ϵ_{high}) close to target E2E delay, and in the others, it improves link utilization while having a similar queuing delay to sL4S. Thus, 1.3% and 1.2% of packets on average in this scenario for cL4S and sL4S, respectively, exceed the target queuing delay, which is the lowest value compared to the other CCs and mostly occurs during rapid link drops (i.e. around 60 seconds on *IETF* pattern). Moreover, we simulated an additional group (i.e. L4S-CC, GCC, and ScreamV2) to compare the performance of aL4S-CC with L4S-CC in the same environment. On average in the experiment, compared to L4S-CC, cL4S and sL4S achieved higher link efficiency rates of 3.4% and 1.3% and reduced the percentage of packets exceeding the target queuing delay by 0.1% and 0.2%, respectively.

B. INTRA-PROTOCOL FAIRNESS

This scenario replicates a real-world network environment, where two homogeneous flows with identical CC are transmitted to the NG-RAN scheduler with the same QoS requirement. We utilize Jain's fairness [23], measured instantaneously at each 100 ms interval, to evaluate fairness. For all cases, Table 1 presents link utilization, QD percentile 95% and 99%, TDQ(20), TDQ(55), TDQ(145), and Jain's fairness index on average of two traffics on *IETF* pattern. GCC performs to achieve fair distribution, with Jain's fairness values ranging from 0.97 to 0.98, similar to those of cL4S and sL4S. Although ScreamV2 faces challenges in maintaining robustness for high link utilization and fair distribution, it generally shows a lower delay than GCC. Ultimately, cL4S and sL4S demonstrate the highest link utilization compared to the other methods, while also achieving fair distribution. Specifically, cL4S improved link utilization on average compared to sL4S, L4S-CC, GCC, and ScreamV2 by 0.6%, 1.4%, 1.7%, and 8.3%, respectively. Besides, as the same behavior shown in the inter-protocol scenario above, the results of aL4S-CC prove to transmit the stable stream with low latency.

C. SINGLE FLOW

This scenario analyzes how stable streaming is transmitted with coping abruptly dropping or static link patterns in the absence of other traffic. Fig. 5 illustrates the received rate and queuing delay measured over the time for a single flow sent

TABLE 2. **Results on cL4S, sL4S, L4S-CC, GCC, and ScreamV2 Under Single Flow**

		Multimodal XR			XR Cloud			XR Conference				
	Metrics	cL4S	sL4S	L4S-CC	cL4S	sL4S	L4S-CC	cL4S	sL4S	L4S-CC	GCC	ScreamV2
Case 1	Link Utilization (%)	84.7	83.6	69.1	86	85.6	69.6	86.6	85.5	70.6	80.9	40.6
	QD Percentile 95% (ms)	2	1	1	2	1	1	3	1	1	45	0
	QD Percentile 99% (ms)	8	6	4	8	6	4	10	11	5	95	14
	TDQ(20) (seconds)	0.2	0.6	0.5	-	-	-	-	-	-	16.3	1
	TDQ(55) (seconds)	-	-	-	0.5	0.2	0.3	-	-	-	6.7	0
	TDQ(145) (seconds)	-	-	-	-	-	-	0.6	0.7	0.2	1.2	0
Case 2	Link Utilization (%)	82.1	82	72.1	86.3	82.4	80.6	86.6	85.4	81.2	80.5	42
	QD Percentile 95% (ms)	5	7	4	7	5	5	9	6	7	51	1
	QD Percentile 99% (ms)	12	15	10	13	14	17	19	15	17	99	26
	TDQ(20) (seconds)	0.5	0.6	0.5	-	-	-	-	-	-	16.8	2.4
	TDQ(55) (seconds)	-	-	-	0.3	0.4	0.3	-	-	-	6.9	0.2
	TDQ(145)(seconds)	-	-	-	-	-	-	0.6	0.5	0.5	1.3	0
Case 3	Link Utilization (%)	60.2	64.1	70.8	69	68.3	72.8	80.2	75.7	74	81.4	53.2
	QD Percentile 95% (ms)	18	20	24	22	24	28	35	24	30	57	20
	QD Percentile 99% (ms)	37	42	45	48	46	53	67	48	58	110	47
	TDQ(20) (seconds)	7.2	8.1	12	-	-	-	-	-	-	17.2	9
	TDQ(55) (seconds)	-	-	-	0.9	0.7	1.6	-	-	-	8.4	1.2
	TDQ(145) (seconds)	-	-	-	-	-	-	0.7	0.5	0.6	1.2	0

with each CC on *IETF* pattern targeting case 2 of XR cloud gaming service, and for all cases, Table 2 shows the metrics with the same structure as Table 1 except Jain's fairness. During a sudden network drop, observed around 60 seconds in Fig. 5, ScreamV2 shows a minor spike originating from a low bitrate. Conversely, although aL4S-CC cannot completely eliminate the latency spike caused by the sudden drop due to high bandwidth, it alleviates prolonged latency spikes by swiftly reducing the rate through (12). In the case of a sudden increase in link availability, observed around 40 seconds, aL4S-CC enhances link efficiency by the rapid probing in (5). In general, GCC has constant results for all cases as shown in Table 2 because it doesn't consider the ECN marking, which leads to maintaining 81% link utilization on average with slight deviation. Also, it can be seen from the results that GCC suffers from a slower reaction against congestion, which leads to a higher number of latency spikes. For ScreamV2, although the rate reduction followed by aggressive increases causes link utilization decreases and latency spikes, link utilization is improved in case 3. Despite L4S-CC achieving low latency, the link efficiency varies depending on the ECN mechanism, as elaborated in Section II. On the other hand, cL4S and sL4S have proven to provide stable streaming without suffering from latency spikes by achieving, on average for all cases, high link utilization of 82% and 81%, respectively, while having low 95% and 99% percentile of the queuing delay as 10 ms and 23 ms at cL4S and 8 ms and 20 ms at sL4S.

VI. CONCLUSION

This paper presents a congestion control algorithm for a cloud-based streaming system, referred to as aL4S-CC. The presented algorithm utilizes the CE-marking ratio, PDV, and ECN marking scheme to react to congestion while aiming at not surpassing a given target E2E delay for different services and keeping a high link utilization, as well as a good performance with respect to fairness and competitiveness. The proposed approach has been evaluated considering the QoS required by immersive services, and its result has shown that cL4S, utilizing the ECN marking scheme including the marking mechanism and delay thresholds used for applying ECN marking, outperforms sL4S which does not incorporate this scheme. The performance of the proposed algorithm has been compared with our previous work and the well-known algorithms GCC and ScreamV2 in different scenarios. When the traffic is simultaneously transmitted with heterogeneous algorithms in the same link, aL4S-CC has demonstrated higher link utilization compared to the others while achieving the fewest latency spikes, ensuring low latency. Besides, aL4S-CC has proven to have the ability to provide stable streaming on single-flow and superior performance with respect to fairness on intra-protocol traffic, showing a Jain's fairness value of 0.97 or more in all cases while still maintaining the high link utilization and low latency.

REFERENCES

[1] Microsoft, "Microsoft project xcloud," Microsoft Corporation, Accessed: May 10, 2024. [Online]. Available: https://developer.microsoft.com/en-us/games/products/project-xcloud/
[2] Meta, "Horizon workrooms," Meta Platforms, Inc., Accessed: May 10, 2024. [Online]. Available: https://forwork.meta.com/horizon-workrooms/
[3] Nvidia, "Nvidia cloudxr: Streaming for extended reality," Nvidia Corporation, Accessed: May 10, 2024. [Online]. Available: https://www.nvidia.com/en-us/design-visualization/solutions/cloud-xr/
[4] M. Abdallah, C. Griwodz, K.-T. Chen, G. Simon, P.-C. Wang, and C.-H. Hsu, "Delay-sensitive video computing in the cloud: A survey," *ACM Trans. Multimedia Comput., Commun., Appl.*, vol. 14, no. 3s, pp. 1–29, 2018.
[5] "System architecture for the 5G system (5Gs)," 3GPP TS 23.501, Accessed: May 10, 2024. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3144
[6] B. Briscoe, K. D. Schepper, M. Bagnulo, and G. White, "Low latency, low loss, and scalable throughput (L4S) internet service: Architecture," RFC 9330, Accessed: May 10, 2024. [Online]. Available: https://www.rfc-editor.org/info/rfc9330
[7] S. Floyd, K. K. Ramakrishnan, and D. L. Black, "The addition of explicit congestion notification (ECN) to IP," RFC 3168, Accessed: May 10, 2024. [Online]. Available: https://www.rfc-editor.org/info/rfc3168
[8] S. Gül et al., "Cloud rendering-based volumetric video streaming system for mixed reality services," in *Proc. 11th ACM multimedia Syst. Conf.*, 2020, pp. 357–360.

[9] "Enabler for immersive real-time communication," 3GPP TS 26.113, Accessed: May 10, 2024. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=4041

[10] J. Son, Y. Sanchez, C. Hampe, D. Schnieders, T. Schierl, and C. Hellge, "L4S congestion control algorithm for interactive low latency applications over 5G," in *Proc. IEEE Int. Conf. Multimedia Expo*, 2023, pp. 1002–1007.

[11] G. Carlucci, L. De Cicco, S. Holmer, and S. Mascolo, "Analysis and design of the google congestion control for web real-time communication (WebRTC)," in *Proc. 7th Int. Conf. Multimedia Syst.*, 2016, pp. 1–12.

[12] I. Johansson and M. Westerlund, "Self-clocked rate adaptation for multimedia," Internet Engineering Task Force, Internet-Draft draft-johansson-ccwg-rfc8298bis-screamv2-00, Accessed: May 10, 2024. [Online]. Available: https://datatracker.ietf.org/doc/draft-johansson-ccwg-rfc8298bis-screamv2/00/

[13] V. Jacobson, "Congestion avoidance and control," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 18, no. 4, pp. 314–329, 1988.

[14] L. S. Brakmo, S. W. O'malley, and L. L. Peterson, "TCP vegas: New techniques for congestion detection and avoidance," in *Proc. Conf. Commun. Archit., Protoc., Appl.*, 1994, pp. 24–35.

[15] S. Ha, I. Rhee, and L. Xu, "Cubic: A new TCP-friendly high-speed TCP variant," *ACM SIGOPS Operating Syst. Rev.*, vol. 42, no. 5, pp. 64–74, 2008.

[16] V. Arun and H. Balakrishnan, "Copa: Practical {Delay-Based} congestion control for the internet," in *Proc. 15th USENIX Symp. Netw. Syst. Des. Implementation*, 2018, pp. 329–342.

[17] M. Dong, Q. Li, D. Zarchy, P. B. Godfrey, and M. Schapira, "{PCC}: Re-architecting congestion control for consistent high performance," in *Proc. 12th USENIX Symp. Netw. Syst. Des. Implementation*, 2015, pp. 395–408.

[18] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-based congestion control: Measuring bottleneck bandwidth and round-trip propagation time," *Queue*, vol. 14, no. 5, pp. 20–53, 2016.

[19] G. Zeng et al., "Congestion control for cross-datacenter networks," *IEEE/ACM Trans. Netw.*, vol. 30, no. 5, pp. 2074–2089, Oct. 2022.

[20] X. Sun, Z. Wang, Y. Wu, H. Che, and H. Jiang, "A price-aware congestion control protocol for cloud services," *J. Cloud Comput.*, vol. 10, pp. 1–15, 2021.

[21] H. Zhang et al., "Loki: Improving long tail performance of learning-based real-time video adaptation by fusing rule-based models," in *Proc. 27th Annu. Int. Conf. Mobile Comput. Netw.*, 2021, pp. 775–788.

[22] T. Dai, X. Zhang, and Z. Guo, "Learning-based congestion control for internet video communication over wireless networks," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2018, pp. 1–5.

[23] R. K. Jain, D.-M. W. Chiu, and W. R. Hawe, *A Quantitative Measure of Fairness and Discrimination*, vol. 21. Hudson, MA, USA: Eastern Res. Laboratory, Digit. Equip. Corporation, 1984.

[24] T. Høiland-Jørgensen, P. McKenney, D. Taht, J. Gettys, and E. Dumazet, "The flow queue CoDel packet scheduler and active queue management algorithm," RFC 8290, Accessed: May 10, 2024. [Online]. Available: https://www.rfc-editor.org/info/rfc8290

[25] K. D. Schepper, B. Briscoe, and G. White, "Dual-queue coupled active queue management (AQM) for low latency, low loss, and scalable throughput (L4S)," RFC 9332, Accessed: May 10, 2024. [Online]. Available: https://www.rfc-editor.org/info/rfc9332

[26] C. Perkins, "Sending RTP control protocol (RTCP) feedback for congestion control in interactive multimedia conferences," RFC 9392, Accessed: May 10, 2024. [Online]. Available: https://www.rfc-editor.org/info/rfc9392

[27] "Webrtc for unity," Unity Technologies ApS, Accessed: May 10, 2024. [Online]. Available: https://github.com/Unity-Technologies/com.unity.webrtc

[28] "Self-clocked rate adaptation for multimedia," Ericsson, Accessed: May 10, 2024. [Online]. Available: https://github.com/EricssonResearch/scream/

[29] "Service requirements for the 5G system," 3GPP TS 22.261, Accessed: May 10, 2024. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3107

[30] "Traffic models and quality evaluation methods for media and XR services in 5G systems," 3GPP TS 26.926, Accessed: May 10, 2024. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3890

[31] "Jens linux kernel modules," Tarent solutions GmbH, Accessed: May 10, 2024. [Online]. Available: https://github.com/tarent/sch_jens/tree/master/janz

[32] Z. Sarker, V. Singh, X. Zhu, and M. A. Ramalho, "Test cases for evaluating congestion control for interactive real-time media," RFC 8867, Accessed: May 10, 2024. [Online]. Available: https://www.rfc-editor.org/info/rfc8867